COVER SHEET

Using Camera Tilt to Assist with Localisation

Trevor Taylor, Shlomo Geva, Wageeh W. Boles
Queensland University of Technology
Brisbane, Australia
{T.Taylor, S.Geva, W.Boles}@qut.edu.au

Abstract

Digital cameras are often used on robots these days. One of the common limitations of these cameras is a relatively small Field of View. Consequently, the camera is usually tilted downwards in order to see the floor immediately in front of the robot. With the camera tilted, vertical edges no longer appear vertical in the image. This feature can however be used to advantage to discriminate amongst straight line edges extracted from the image when searching for landmarks. It might also be used to estimate angles of rotation and distances moved between successive images in order to assist with localisation.

Keywords: computer vision, monocular camera, camera tilt, perspective, visual localisation

1 Introduction

The price of digital cameras has plummeted in recent years due to the popularity of web cameras and cameras in mobile phones. This has provided a useful sensor for robots.

These cheap cameras often have poor optics and a limited Field of View (FOV) – as little as 40–60°. This is a problem if the robot needs to use the camera to locate obstacles in its path because the camera needs to be tilted downwards. One side-effect is that vertical edges no longer appear vertical in the image.

This paper addresses the issue of camera tilt and attempts to draw some positive benefit from it. The term "tilt" in this context means angling the camera downwards towards the floor so that the principal axis of the camera is below the horizontal. This might also be referred to as pitch (as opposed to yaw or roll) and sometimes the camera azimuth angle.

There are alternatives to using a single camera with a small FOV in order to avoid camera tilt, such as stereo vision, wide-angle lenses or even panoramic or omni-directional cameras. However, these solutions are more expensive and introduce the problems of complex calibration and/or dewarping of the image. Therefore, we persist with a single camera.

In this paper, the term "vertical" is used to refer to any edge of a real world object that is vertical, e.g. a door frame or the corner where two walls meet. This is distinct from a "horizontal" which refers to an edge that is horizontal in the real world, e.g. the skirting board where a wall meets the floor. These vertical and horizontal edges do not usually appear as such in a camera image, especially when the camera is tilted.

Note that for a robot moving in two dimensions on a horizontal ground plane, i.e. the floor, the location of a vertical can be specified using only two coordinates.

1.1 Related Work

Our objective is for a robot to navigate in an indoor environment using only monocular vision and build a map as it goes. Several algorithms have emerged in recent years for Simultaneous Localisation and Mapping (SLAM) using vision.

In this paper we do not present a complete visual SLAM algorithm, however we outline the theoretical underpinnings of a new approach to localisation that takes advantage of camera tilt.

The SIFT (Scale-Invariant Feature Transform) algorithm has been used for visual SLAM by several researchers, including the original developers [1]. Because SIFT features have distinctive signatures, no attempt is made to extract lines from images.

In [2], the approach is also to identify significant features in images and then determine location using an image retrieval procedure. To make this robust, they incorporate Monte Carlo Localisation. Although they rely on an algorithm for calculating the signature of selected image features, they also note that features such as lines are useful in general.

Conversely, a Structure from Motion (SFM) approach has also been used. This relies on developing a model of the real world by solving sets of equations based on point correspondences derived from multiple images. Davison used this approach, and more recently has enhanced his original work by using a wide-angle lens [3]. The key advantage claimed is that the wider field of view allows landmarks to remain in view for longer and therefore contribute to more matches. However, it is pointed out that a special model is required for the camera because it does not conform to the conventional perspective projection.

Eade and Drummond [4] apply the particle filter approach of FastSLAM to visual SLAM using a single camera. As in the previous methods, the

selected features are based solely on how distinctive they are, not on their significance in the real world.

Visual odometry has been demonstrated using a commercial web camera [5]. This relies basically on optic flow. The intention in this case is obstacle avoidance and reliable odometry, but not the repeatable identification of landmarks. Therefore the algorithm does not extract lines.

We take a geometric approach to processing images to identify vertical edges. These edges are important because they represent significant features in the environment. This approach follows the seminal work of Faugeras [6] on 3D vision. However, in his work he used matrix notation and dealt with the general case. We have taken specific cases with simple solutions that seem to have gone unnoticed.

1.2 Background

By convention, the coordinate system used for a camera has its origin at the focal point of the camera. figure 1 shows the side view of a robot with a camera mounted on top of it. The *X*-axis is into the page and the negative *Z*-axis is along the camera's principal axis (for a right-hand coordinate system).

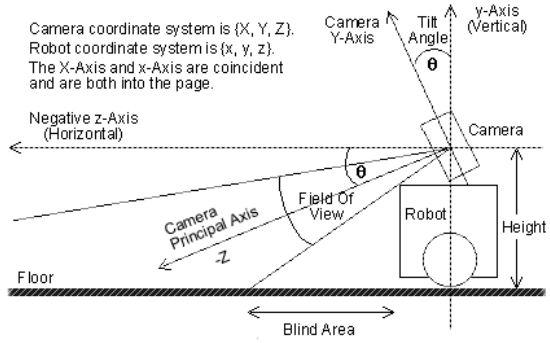

Figure 1: Camera coordinate system.

The diagram shows that the camera is tilted. It also shows the vertical Field of View. (The horizontal FOV can be calculated based on the aspect ratio of the camera.) Notice that there is a blind area in front of the robot. The choice of camera tilt angle is therefore a compromise between visible range and being unable to see small obstacles in the immediate vicinity.

The robot coordinate system {*x, y, z*} has its origin at the same point as the camera coordinate system {*X, Y, Z*}. It would make more sense to have the origin of the robot coordinates on the floor, but only the *x* and *z* coordinates are of interest because the robot moves in two dimensions. Therefore, the height off the floor (in the *y* direction) is not important except to the extent that it affects the size of the blind area.

Ignore the camera tilt for now. Start with the standard pin-hole camera equations. (Refer to a textbook on Computer Vision, such as [7] page 20.)

$$u = \frac{fX}{Z}, \quad v = \frac{fY}{Z} \tag{1}$$

where (*u, v*) are the coordinates of a pixel in the image (assuming the image origin is at the centre) of a real-world point (*X, Y, Z*) expressed in the camera coordinate system, and *f* is the camera focal length.

By using an edge detector and a line detector, e.g. the Canny Edge Detector and Probabilistic Hough Transform in the Intel OpenCV Library [8], it is possible to obtain a set of lines from an image.

The slope of any given line in an image (which is a 2D space) can be obtained from two points on the line, (u_1, v_1) and (u_2, v_2), as follows:

$$m = \frac{v_2 - v_1}{u_2 - u_1} \tag{2}$$

where *m* is the slope of the line.

If the actual angle is required, then this can be calculated as arctan(*m*). Note that the nature of the tangent function is such that *m* approaches infinity as the line approaches vertical, which can create computational problems. An alternative would be to use the inverse slope, which would move the singularity to lines of zero slope – well outside our range of interest.

Substitute into equation (2) after using equation (1) to determine the image coordinates for two real-world points, p_1 and p_2. After a small amount of simplification the result is:

$$m = \frac{Y_2 Z_1 - Y_1 Z_2}{X_2 Z_1 - X_1 Z_2} \tag{3}$$

This equation describes the slope of a line between two points as it appears in the image.

It is worth noting at this stage that the focal length has disappeared from the equation. Also, it is irrelevant which point is chosen as p_1.

2 Finding Vertical Edges

Vertical edges are important as landmarks, especially in man-made environments. These features can be useful for localisation of a robot after small motions.

For a camera that is not tilted, finding vertical edges is trivial because the edges are vertical in the image. However, for a tilted camera this is no longer the case. Furthermore, the slope of the "vertical" edge in the image is dependent on where the edge is physically located with respect to the camera.

2.1 The Effect of Camera Tilt

Consider the case when the camera is tilted from the horizontal as shown in figure 1. This is a rotation about the *X*-axis of the camera by an angle of θ.

Henceforth it is assumed that this angle is known. If necessary, it can be obtained from the extrinsic parameters of the camera via a calibration procedure.

It might be tempting to think that a transformation of the image could undo the effects of the camera tilt. However, when a camera moves the images are not related by a projective transformation unless all of the points in the image are coplanar [9]. (This principle is exploited in stereo vision to detect depth disparities.) In other words, it is impossible to correct for projective distortion across a whole image without knowing the locations of all the objects in the image.

Before images can be processed, especially where straight lines are involved, the effects of camera distortion must be removed, i.e. the camera needs to be calibrated, e.g. using [10]. This is particularly important for cheap lenses which often suffer from some sort of radial or barrel distortion.

figure 2 shows objects viewed from a tilted camera. The left-hand image is the raw image from the camera, and the right-hand one is after correction for lens distortion. It is clear that the vertical edges are not vertical, nor parallel to each other. In fact, if all the "verticals" were extrapolated they would meet at a vanishing point which is well outside the image.

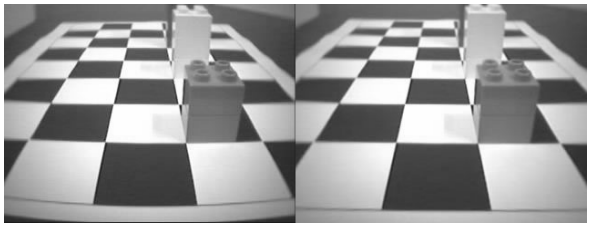

Figure 2: Variations in slope of vertical edges.

We would like to use the information in the image to assist in identifying the location of obstacles. It is more convenient for the robot to use its own coordinate system, rather than camera coordinates, because the robot's x and z coordinates relate directly to distances that can be measured across the ground.

The effect of the camera tilt is that the x coordinates of all points in robot space remain the same in camera space, i.e. $x = X$. However, the y and z coordinates must be transformed. The new coordinates can easily be calculated as explained below.

For notational convenience, let $c = \cos(\theta)$ and $s = \sin(\theta)$. The rotation about the X-axis is:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \tag{4}$$

In terms of robot coordinates, the rotated coordinates of two points can be expressed in camera space as:

$$\begin{aligned} p_1 &= \left(x_1,\ y_1 c - z_1 s,\ y_1 s + z_1 c\right) \\ p_2 &= \left(x_2,\ y_2 c - z_2 s,\ y_2 s + z_2 c\right) \end{aligned} \tag{5}$$

If the two points were selected so that they were both on a vertical edge in the real world, then in the robot coordinate system $x_1 = x_2$ and $z_1 = z_2$, i.e. only the y coordinate varies. Therefore no subscript is used on x or z in the following equations.

Using a similar approach to that used to obtain equation (3), substitute these new coordinates from (5) into equations (1) and (2) and simplify:

$$m = \frac{z\left(y_2(c^2 + s^2) - y_1(c^2 + s^2)\right)}{x s (y_1 - y_2)} \tag{6}$$

Applying the trigonometric identity $c^2 + s^2 = 1$, the final result is:

$$m = \frac{-z}{x s} \tag{7}$$

This is a very simple result, but it should not be surprising given what we know from our personal experience about how perspective works.

Some conclusions can be drawn from equation (7). Firstly, when $x = 0$ the slope is infinite, i.e. a vertical edge will appear vertical at the centre of the image. This singularity can be avoided by inverting the calculations and using $1/m$ instead. However, the results are then very close to zero near the centre of the image, and are still difficult to compare.

Also, the sign of the slope changes as the x coordinate changes from positive to negative. This means that the vertical edges appear to slope further and further away from vertical the further they are from the centre. This feature can be used as a heuristic to filter out line segments as possible verticals.

And finally, since $\sin(0) = 0$, the slope also becomes infinite if the camera tilt angle is zero, regardless of the values of x or z. In other words, the lines associated with vertical edges are all vertical in the image if the camera is not tilted.

Figure 3 shows a simulation where the camera has a 60° horizontal FOV and is tilted downwards at 30°. Vertical lines have been drawn from the corners of the checkerboard pattern on the floor. The slopes change significantly across the image, as expected.

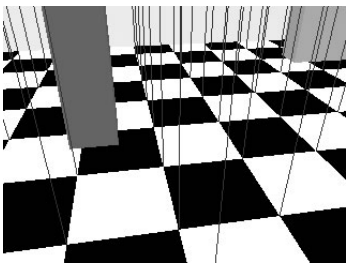

Figure 3: Verticals as viewed by a tilted camera.

The maximum possible slope for a vertical edge is determined by the FOV of the camera and its height above the floor. This sets a limit on the ratio of z to x that is possible for a given camera. For the example shown above, the "worst case" slope is around 66°. Over much of the image the slope will be close to vertical. For instance, most of the verticals in figure 3 have a slope in the range of 70° to 90°.

Another aspect of equation (7) that is apparent in figure 3 is that the equation describes families of lines. For any set of vertical edges with the same ratio of z to x, the slope will be the same. In fact, these lines will be coincident in the image.

Notice that the leftmost edge of the large object on the left in figure 3 almost coincides with one of the vertical lines, even though the respective vertical edges originate from different places on the floor.

It is easy to see that horizontals, e.g. the checkerboard pattern, would be difficult to confuse with the verticals even though the verticals do not appear vertical in the image.

If the point of intersection of the vertical edge with the floor can be determined, then this can be used to obtain the x and z coordinates of the feature using Inverse Perspective Mapping via a simple table lookup [11]. For these coordinates, the slope of the vertical edge can be calculated, thereby confirming whether or not the intersection point is correct.

3 Incremental Localisation

A common problem in computer vision is establishing correspondences between points in images. This is often done on the basis of small patches in stereo vision for instance. In our case of using vertical edges, making the correspondence between two edges in successive frames for a moving camera is relatively straight-forward.

In the discussion that follows we consider two different types of moves that can be performed by a robot with two independently driven wheels. These two motions – translations and rotations – allow the robot to follow any possible 2D path.

3.1 Translations

Faugeras [6] points out (page 283) that it is not possible to determine camera displacement based on the correspondence between lines in two images. However, we add a further constraint in that we only allow our robot to move in a straight line. This reduces the number of degrees of freedom so that there is an analytical solution to the problem.

For a forward motion, i.e. translation along the negative z-axis, the appearance of a given vertical edge will change. It is theoretically possible to use the change in line slope to measure the distance moved, or alternatively confirm the distance moved.

Assume the initial feature coordinates of interest are x_1 and z_1. (The y coordinate is irrelevant for a vertical edge because the edge runs parallel to the y-axis.) After the move, the new coordinates will be x_2 and z_2.

The slope of the line can be measured in the image before and after the move. The two slopes are:

$$m_1 = \frac{-z_1}{x_1 s}, \quad m_2 = \frac{-z_2}{x_2 s} \tag{8}$$

The x coordinate of the edge is not affected by the move, so $x_1 = x_2$. Expressing the equations (8) in terms of x and then setting them equal to each other results in:

$$z_1 = \frac{m_1}{m_2} z_2 \tag{9}$$

If the distance travelled can be measured accurately, e.g. using odometry, then the difference between z_1 and z_2 is known, say $d = z_1 - z_2$.

Substituting into equation (9) and solving for z_1 gives:

$$z_1 = \frac{m_1}{(m_1 - m_2)} d \tag{10}$$

A simple rearrangement of equation (10) allows d to be calculated if z_1 is known (or z_2 for that matter):

$$d = \left(1 - \frac{m_2}{m_1}\right) z_1 \tag{11}$$

At first glance, this looks like a simple way to determine the range of objects based solely on odometry, or conversely to use range information to verify odometry data.

However, in practice odometry is notoriously unreliable. Furthermore, the difference between the two slopes will often be very small and might not be measurable to within an acceptable level of error given the resolution of the imaging device.

Following a similar derivation, it is possible to use a motion in x direction. A sideways motion can have a much larger impact on the slope of vertical edges for nearby objects than forward motion. However, this type of motion is only possible for holonomic robots, not for wheeled robots.

3.2 Rotations

Consider the situation where the camera rotates on the spot, i.e. around the y-axis. This might occur for a camera that can pan, or for a two-wheeled robot that can turn on the spot if the camera is located in the centre of the robot.

Again, for a given vertical edge, let the feature coordinates be x_1 and z_1 and after rotation the new coordinates will be x_2 and z_2.

(These coordinates are related by the radial distance from the camera, which must remain constant. We do not need to use this information, but it is a constraint that should be checked.)

In the robot's view of the world, the angles between the x-axis and vectors drawn to the vertical edge are given by:

$$\tan(\alpha) = \frac{z_1}{x_1}, \quad \tan(\beta) = \frac{z_2}{x_2} \tag{12}$$

The difference between these two angles is the desired angle of rotation, i.e. we want to know $\psi = \beta - \alpha$.

Applying equation (7) to the equations (12), the resulting difference between the two angles, i.e. the amount of rotation, is:

$$\psi = \arctan(m_1 s) - \arctan(m_2 s) \tag{13}$$

(When calculating this difference, allowance must be made for a possible change of sign if the vertical edge moves across the centre of the image.)

4 Examples

The following two examples are for rotations.

The top two images in figure 4 show two different views with a rotation of 20° between them. The bottom two images show the vertical edges detected using the algorithm in this paper as thick lines.

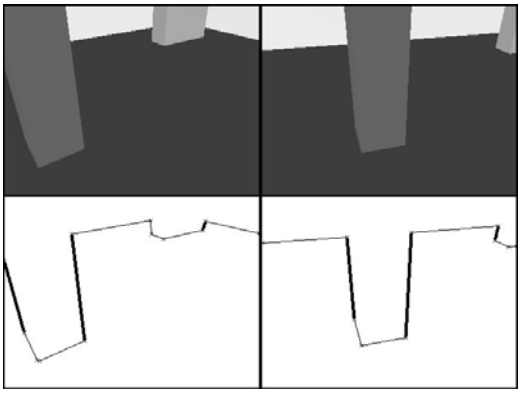

Figure 4: Rotation example 1 using simulator.

There are two significant verticals visible in both images. Calculating the slopes of these lines allows the angle of rotation to be calculated according to equation (13). The results are shown in table 1.

Table 1: Slopes and rotation angles for example 1.

Slope m_1*	Slope m_2	Angle ψ
106.82°	95.16°	20.9°
97.54°	87.54°	19.7°

* Slopes have been converted to angles in degrees because it is easier to verify them visually. This is not part of the calculation.

These estimated rotation angles are very close to the actual value, at least to within a 1 pixel quantization error in the location of the edges in the image.

The next example shows images from an X80 robot. The robot was given three commands to rotate right by 10°. Segmentation of the floor is work in progress, so the edges of the door jamb have been marked manually. The images are shown in figure 5.

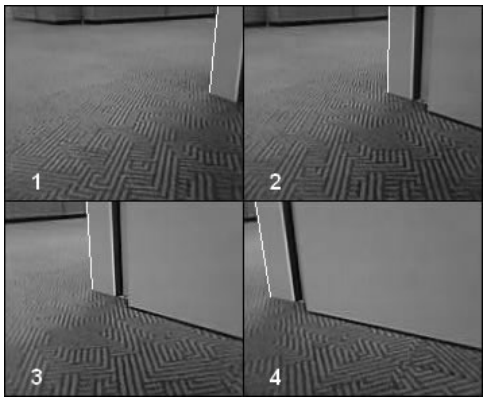

Figure 5: Rotation example 2 from X80 robot.

The camera has a resolution of 176x144 pixels. It was calibrated using the software from [10]. This software provides the camera extrinsic parameters. From these, the tilt angle was calculated to be 18.97°, and the transformation between pixel coordinates and real world coordinates was also determined.

On the X80 robot, the camera is not located at the centre of the robot. Therefore the offset from the centre was included when calculating the actual turn angles based on the camera's extrinsic parameters.

The results are shown in table 2 below. This gives the angle calculated based on the camera extrinsic parameters, δ, and the angle calculated according to equation (13) above, ψ.

Note that in the second image the edge is very close to vertical. An error of one pixel at either end of the edge could result in the calculation of m becoming undefined. The code needs to handle this as 90°.

Table 2: Calculated rotation angles for example 2

Image	Angle δ	Angle ψ
2	10.96°	10.72°
3	10.33°	8.09°
4	10.39°	14.92°

When the robot had completed all the rotations, it had moved slightly to the left. This might account for the poor estimate of the last angle using equation (13) which assumes no movement. The angle calculated using extrinsic parameters, however, is a relative

change and is not affected in the same way. It is hard to say which of the results is actually correct.

4.1 Practical Considerations

Different image processing tools produce different results. When the images are processed using Canny Edge Detection followed by Hough Lines in OpenCV and Matlab, the results are not identical, even when using ostensibly the same parameters.

Also, the Hough Transform, although widely used in computer vision, is a parametric algorithm and it must be tuned to obtain the best results. In particular, bin sizes must be specified for the radial distances and angles of the lines. Even small changes in the parameter for the bin size for line angles can result in missing a line because it does not line up precisely with one of the bin angles. Conversely, using a bin size for the radial distance that is too small can result in a myriad of extraneous lines.

Accurately detecting the slope of the lines is essential for our algorithms to produce accurate results for translations and rotations. However, the approach outlined here is not affected by errors in the absolute location of an edge in the image. Only the slope of an edge is used, which is obtained from the differences between pixel coordinates.

We have not performed a rigorous analysis of variance, but it is possible to place a reasonable limit on the range of errors simply using trial and error.

Detecting edges accurately in a digital image is difficult because of the quantization effects. Although there are algorithms that will locate an edge with sub-pixel accuracy, it is reasonable to assume that there might be an error of ±1 pixel in determining the image coordinates of any edge pixel.

In the worst case the errors at either end of the line will combine in the same direction. By varying the coordinates at each end of a line by one pixel, it was found that the resulting change in the calculated rotation angle ranged from ½ to 2 degrees. This represents an error of around 5-10% for a 20° rotation. This is not much better than the accuracy of the wheel encoders on the X80, so it is of questionable value in this case.

The images in figure 5 were obtained by deliberately positioning the robot so that the door edge would stay in view for three rotations. In general, most edges disappear after only two, so tracking edges for more than one rotation is not always possible, especially if the rotation is more than 10 degrees.

Finally, the resolution of the images used in these examples is typical for cheap web cameras. However, consumer digital cameras now have resolutions up to 8 megapixels. This is over 100 times the resolution we have used. Even assuming only a ten-fold improvement in the accuracy of measuring the slope of lines, it is clear that more accurate results could be obtained with better cameras.

5 Conclusion

This paper has outlined some simple geometric relationships that can be used to recover vertical edges from an image when the camera is tilted. By comparing the slopes of vertical edges in successive images, it is possible to calculate distances moved or angles of rotation. This information can be used to assist with localisation. We are working to apply this to Visual SLAM.

6 References

[1] S. Se, D.G. Lowe and J.J Little, "Vision-based global localisation and mapping for mobile robots", *IEEE Transactions on Robotics*, Vol. 21, No. 3, pp 364-375 (2005).

[2] J. Wolf, W. Burgard and H. Burkhardt, "Robust vision-based localisation by combining an image retrieval system with Monte Carlo localization", *IEEE Transactions on Robotics*, Vol. 21, No. 2, pp 208-216 (2005).

[3] A.J. Davison, Y.G. Cid and N. Kita, "Real-time 3D SLAM with wide-angle vision", *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, CD-ROM (2004).

[4] E. Eade and T. Drummond, "Scalable monocular SLAM", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, NY, USA, Vol. 1, pp 469-476 (2006).

[5] J. Campbell, R. Sukthankar, I. Nourbakhsh and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision", *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp 3421-3427 (2005).

[6] O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA, USA (1993).

[7] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, USA (1986).

[8] Intel Corporation, "Open Source Computer Vision Library " (OpenCV), http://www.intel.com/technology/computing /opencv/, visited 2/8/2005.

[9] R. Hartley, and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd Ed., Cambridge Univ. Press, Cambridge, UK (2003).

[10] J-Y. Bouguet, "Camera calibration toolbox for Matlab", http://www.vision.caltech.edu/bouguetj /calib_doc/, visited on 19/3/2006.

[11] T. Taylor, S. Geva, and W.W. Boles, "Monocular vision as a range sensor," *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, Gold Coast, Australia, CD-ROM, (2004).